UNIVERSIDAD DE GRANADA

Computational intelligence

Pr'actica neural network

Optical character recognition MNIST

course 2019-2020

M' aster in Ingenier'ıa Inform' Attica

Department of Sciences Computaci' and on artificial intelligence

Practice 1

MNIST

The purpose of this is to solve a problem pr'actica pattern recognition using artificial neural networks. Deber'a evaluate the use of several types of neural networks to solve a problem OCR recognition d'ıgitos manuscripts of the database MNIST (http://yann.lecun.com/exdb/mnist/).

1 Implementaci'on

To perform this pr' Actica can choose to implement algorithms yourself it deems appropriate training or use as a basis for many libraries that implement these algorithms and est'an available p' ublicamente, although the decisions about can influence in their final caci'on qualify.

Although the mayor'ıa implementaci'on sequential algorithms neural network training is not particularly complex, you can be found at m'Internet ulti- ples libraries that implement some of the algorithms described in class so efficient. Such implementations available for m' ultiple languages programaci' on, usually dise~ nothings to take advantage of the various n' ucleos a microprocessor ac- tual or ability to c ' omputo a GPU, if you have one available. If, Instead of using a language programaci'on of prop' General bear, prefers to use a matem' tool atica, tambi'en can find *toolboxes* Matlab or packages in R pr' acticamente for any type of neural network you want to evaluate. However, the use of libraries provided by third tambi'en has some drawbacks that must be taken into account. First, no one guarantees that implementaci'on est'e available completely free of errors and, although as'ı is, it may prove di ffi cult to adapt the implementaci' available on our needs concrete when we want to perform a particular experiment.

> IMPORTANT: Use third-party implementations without making the corresponding attributes is constitutive of plagiarism, a crime against intellectual property that carries a suspense autom' Penthouse in the subject.

On the computation of the gradient

When making your implementaci'on, we recommend using a strategy of TDD type [*test-driven development]* or, at least, perform unit tests to ensure that the C' gradient computation is performed correctly. To perform this comprobaci'on, remember the de fi nition of the derivative:

$$\frac{d}{d\theta} J(\theta) = \lim_{\to 0} \frac{J(\theta + ?) - J(\theta -)}{2}. \tag{1.1}$$

For any value θ, You can approximate the derivative using a value of epsilon ( ) peque~ not, eg 10^{-4} (a value too peque~ podr'ıa not cause rounding errors, so you should not rush too in this regard):

$$g(\theta) \approx \frac{J(\theta + ?) - J(\theta -)}{2}. \tag{1.2}$$

How θ It is a vector of par' ameter, not a simple n' Real UMBER, we will have committed bar the gradient is calculated correctly for each θ_i. For each vector par'ametros θ, evaluate $g_i(\theta)$ as an approximation of $\frac{\partial}{\partial \theta_i} J(\theta)$. We can define $\theta^{(i+)} = \theta + ? \vec{e}_i$, where $\vec{e}_i$ It is a vector with a one in the posici' on ith and zeros dem'as:

$$\vec{e}_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \tag{1.3}$$

So $\theta^{(i+)}$ is similar to θ, unless the ith component has increased . In the same way, we can get $\theta^{(i-)} = \theta - \vec{e}_i$, vector θ with its ith component reduced . With this we can veri fi car num'ericamente gradient for each par'ametro:

$$g_i(\theta) \approx \frac{J(\theta^{(i+)}) - J(\theta^{(i-)})}{2}. \tag{1.4}$$

S'olo we have to repeat the C' computation for different values of θ, with the objective of check that the difference between the values calculated by our implementaci' on and num'ericamente approximate values do not differ too much.

Another interesting alternative is the utilizaci'on of t'ecnicas of autom' diferenciaci'on atica (https://en.wikipedia.org/wiki/Automatic_differentiation). However, implementaci'on of such t'ecnicas I s'olo can compensate for larger projects. B'asicamente, the diferenciaci' on autom' atica is responsible for calcular the gradient of a function for us, so that potential errors are avoided implementaci' on that, otherwise, podr'ıan get to go unnoticed.

2 results An'alisis

When solving a problem classi fi caci'on, the classi fi er is trained with the training set. In this case, the training set contains examples labeled 60000. Examples of training are im'

p'ıxe- standard 28x28 thumbnails

and they are in the fi le train-images-idx3-ubyte, while for Examples labels can be found in the fi le train-labels-IDX1-ubyte.

Evaluate the quality of a classi fi er using the same data that you train may be enga~

noso, why you have a separate set of data,

that is not used during training. Said test assembly, stored in the same format as the training set can be found in the fi les

t10k-images-idx3-ubyte (im' thumbnails) and t10k-labels-IDX1-ubyte (tags).

While almost any t'ecnica of learning autom' attic can get results exceptional on the training set, t'ecnicas not all are equally good when it comes to the test set. Some t'ecnicas of sobreaprenden learning (excess adapt to the training set with the model trains) and then not generalize properly, so they get poor results on different data sets to the training set.

To evaluate the results of the pr'actica use the error rate on the test set (the n'

umber of test set instances that our neural network does not

classifies correctly). As a guideline Inventors, these are the results that deber'ıa get some concrete models of neural networks artificial:

- simple neural network with an input layer and an output layer softmax type: 7.8% error on the test set and 5.6% error on the training set (training time required: about one minute using a implementaci'

  on an interpreted language like Matlab).

- multilayer neural network with one hidden layer 256 log'ısticas units and an output layer softmax type: 3.0% error on the test set and 0.0% error on the training set (training time required: about four minutes using a implementaci'on in an interpreted language as Matlab).

- convolutional neural network trained with stochastic gradient descent: 2.7% error on the test set (training time required: about 13 minutes using a implementaci'on in an interpreted language as Matlab).

- "Deep learning" using pre-training to extract autoencoders caracter'ısticas of im'

  t'ecnicas thumbnails using unsupervised neural network and single

  with a layer of softmax type: from 1.8% to 2.2% error on the test set (training time required: about twenty minutes using a implementaci'

  on an interpreted language like Matlab).

The proposal by Yann LeCun network, convolutional type and m' ultiple hidden layers, able to reduce the error rate to 0.82% (82 errors 10000 examples set test). Using an algorithm of "deep learning" m' so as fi sticado, the error can reduced to 0.35% (35 errors on 10000). combining m' ultiple neural networks You can reduce a' one M'as error, to 0.23% (s' olo 23 errors over 10000). In the p'Agina MNIST website (http://yann.lecun.com/exdb/mnist/) you can find the results that have been obtained with many t'ecnicas classi fi caci' on, including several types of artificial neural networks.

3 Documentaci'on and delivery pr'actica

- Train different neural networks on artificial training set MNIST and eval' ue the results obtained using the test set. Env'ıe the results that are obtained at the p'trav'es Agina web enabled purpose (https://goo.gl/xiXVSK). You can send your results many times as desired, s' olo is tendr' to consider the best result obtained.

  NOTE: The env'ıo results must be performed using the same direcci' on email with whom he appears registered in DECSAI and must include all necessary par'ametros to replicate the experiment, including topolog'ıa of the neural network used (n' umber of layers, layer neurons, neurons type ...) and the algorithm Training used to adjust the weights of the network (with combinaci' on par'ametros particular there is used).

- Create a memory in which is collected experimentaci'on performed during elaboraci' on this pr'actica and cos phi gr'a considered appropriate to illustrate the results obtained. Memory deber' to be delivered in PDF format and ir' to accom- pa¬nothing of all fi les for the various implementations made (and the corresponding links to external libraries that have employed). Memory delivery and PDF fi les c'odigo in a ZIP deber' to reali- zarse access to trav'es identi fi ed DECSAI (https://decsai.ugr.es/) before the December 8, 2019 at 23:59.

Evaluaci' on the pr'actica

- 30% work implementaci'on points:
  - 10% m' aximum if you use a implementaci' on external algorithms learning neural network (0% if it merely replicate experiments alg' a tutorial available on the Internet).
  - 30% at most if you develop your own implementaci' on algorithms Learning neural networks.
- 40% by memory pr' Actica (documentaci' on experiments and tests An 'done with his Alisis results):
  - 20% by descripci'on its implementaci' and on the algorithms used realizaci'on during the pr' Actica.
  - 20% by descripci'on of the experiments you made to establish par'ametros your neural network and An ' parsing the results obtained.
- 30% by the results obtained (using a linear ranking fi nest on the error rates obtained on the test set).